

WinFS Replication

Doug Terry
Microsoft Research Silicon Valley

June 2003



WinFS Synchronization & Caching Team

Mission: *Provide a state-of-the-art synchronization platform integrated into and leveraging WinFS in support of diverse applications requiring peer-to-peer replication for data sharing, high-availability, and offline access.*

Irena Hudis – Product Unit Manager
Lev Novik – Architect



Talk Outline

This morning:

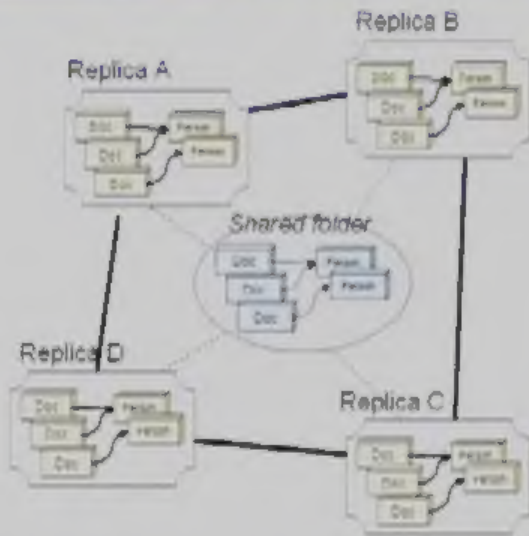
- Goals and requirements
- Change units, versions, knowledge
- Update propagation
- Conflict detection and resolution

This afternoon:

- Filtered sync
- Meta-data cleanup
- Open Issues



Goal: Replication of WinFS Folders



WinFS Synchronization Requirements

- Allow multi-master updates
- Support peer-to-peer topologies
- Operate at the level of WinFS Schema
- Permit configuration of “what” and “when” to sync
- Detect and handle conflicts during synchronization
- Accommodate varying network characteristics: intermittent connections, slow networks, offline
- Provide out-of-the-box “zero admin” manageability, resilience, and ease of use
- Guarantee convergence, even in “unmanaged” environments



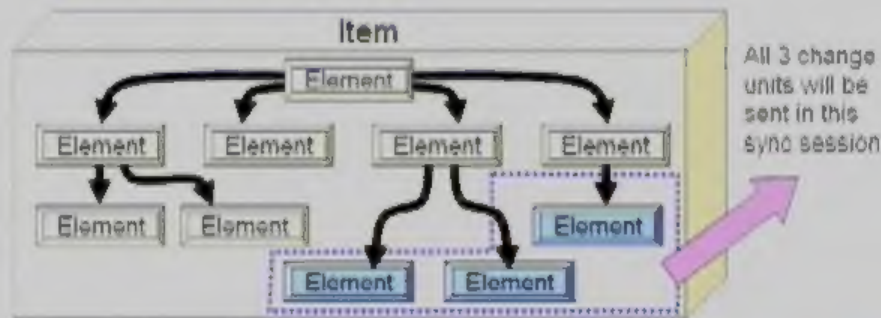
WinFS Synchronization – Approach

- Read-anywhere/update-anywhere model
- Pair-wise reconciliation between replicas
- User-defined topology and schedule
- "Net Changes" of WinFS items/folders are conveyed, not individual actions
- Adapters for replication with non-WinFS machines, services, and devices (*not discussed in this talk*)



Items vs. Change Units

- Item = granularity of consistency
- Change unit = granularity of update, propagation, and conflict detection
 - elements or fields of elements



Versions of Changes

- Each change has a globally-unique id (version)
- Version = ID of the replica making the change + replica-specific number
 - Replica IDs are GUIDs
 - Replica-specific number is ever-increasing at the replica
 - e.g. A5, B2, C7
- Version of change unit is version of last change applied to that element
 - e.g. 10: A5
- Versions only used by synchronization protocol; not visible to WinFS users



Knowledge for Update Propagation

- **Knowledge** = concise description of the set of changes received by a replica
 - knowledge depends only on local database
 - not pair-specific: not "what I have received from you"
- **When requesting changes, specify**
 - Current knowledge (i.e. "don't send these")
 - Filter (i.e. "only interested in these")
- **When enumerating changes**
 - Return change units and tombstones not covered by requester's knowledge
- **When conveying changes, specify**
 - Version of the change
 - The change itself
 - "Made with" knowledge – for conflict detection
 - "Learned" knowledge – not to send them again



Knowledge Abstractly

Main operations on knowledge:

- Test if a given knowledge covers a given change
 - "Does he already know about this change?"
- Add a change version to one's knowledge
 - "I just learned of a new change."
- Merge one knowledge with another to produce combined knowledge
 - "I knew this and he knew that, and now we both know this."



Knowledge Implementation

- Version vector = set of (replica GUID, replica-specific number) pairs
- Semantics: "all versions authored by this replica up to this number"
- Exceptions needed for changes received out-of-order
- Includes every replica that ever made a change

Protocol Using Knowledge

Machine A

E: 44
J: B3
K: C5
L: 00

Machine B

E: 34
J: C2
K: B6
L: 00

Knowledge

A4 B3 C5 D0

Knowledge

A3 B5 C4 D0

Protocol Using Knowledge

Machine A

```
1 request changes
2 ,base = -4 b - 100
3
4 k: --
5 | A2
```

Knowledge

A4 B3 C6 D0

Machine B

```
1
2
3 k: B5 2
4 | A2
```

Knowledge

A1 B0 C4 D1

Protocol Using Knowledge

Machine A

I
K: A2

requestChanges
base = A4 B3 C6 D0

conveyChanges
bundle 1
C2

Knowledge

A4 B3 C6 D0

Machine B

I
K: B5 A2

Knowledge

A4 B5 C4 D2

Protocol Using Knowledge

Machine A

Machine B

requestChanges
base: A4 B3 C5 D0

k
|
A2

k
|
A2

conveyChanges
bundle 1
C2

Knowledge

Knowledge

A4 B3 C5 D0
+D2

A4 B3 C4 D2

Protocol Using Knowledge

Machine A

J
k
l

requestChanges
hash = A4 B3 C6 D0 + D2

conveyChanges
bundle 1
C

Knowledge

A4 B3 C6 D0
+D2

Machine B

J
k
l
A2

Knowledge

A1 B5 C4 D2

conveyChanges
bundle 1
k B5 A

Protocol Using Knowledge

Machine A

1
2
k
P A2

requestChanges
base = A4 B3 C5 D0

Machine B

1
2
k
P A2

conveyChanges
bundle 1
C2

Knowledge

A4 B3 C5 D0
+D2
+B5

Knowledge

A4 B3 C4 D2

conveyChanges
bundle 1
k B5 &

completeBundle
bundle 1
messages 2
earned = B4 C4

Sync Protocol : Things Not Shown

- **Conflict detection and resolution**

- Detecting concurrent updates
- Merging or automatic resolution

- **Batching**

- Batch updates and queries
- Batch commit

- **Topology optimizations**

- Replication factor, replication topology

- **Filtering**

- Replication filters
- Query filters

- **Meta-data cleanup**

- Replication metadata
- Query metadata

Conflict Definition

- Two changes conflict iff
 - (1) the semantics of the changes are incompatible
 - (2) the changes were performed concurrently
(without knowledge of each other)

Conflict Detection using Knowledge

- "Made with" knowledge = knowledge of replica when change was accepted



- Changes C_1 and C_2 are concurrent iff
 - (1) C_1 version not in C_2 madeWithKnowledge and
 - (2) C_2 version not in C_1 madeWithKnowledge

Protocol with Conflict Handling

Machine A

```

1  ... m E ...
2  ...
3  ...
4  ... m E ...
5  ... m E ...

```

Machine B

```

1  ... m E ...
2  ... E ...
3  ...
4  ... m E ...
5  ... m E ...

```

Knowledge

... m E ...

Knowledge

... m E ...

Conflict log

Conflict log

Protocol with Conflict Handling

Machine A

| | | | |
|---|---|---|---|
| l | h | F | 1 |
| j | h | 1 | |
| k | h | E | |
| l | h | 1 | 1 |

request changes
base = add E to L

Machine B

| | | | |
|---|---|---|---|
| l | h | E | 1 |
| j | h | F | 1 |
| k | h | E | 1 |
| l | h | 1 | 1 |

Knowledge

h E 1

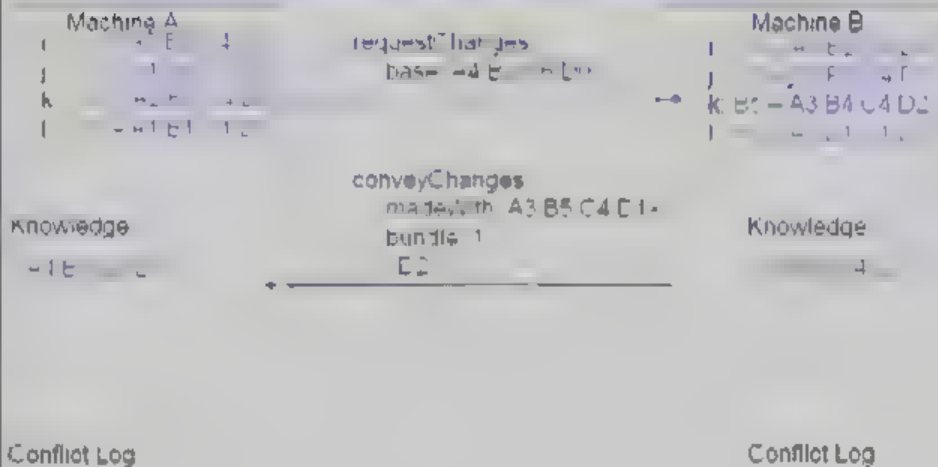
Knowledge

h E 1

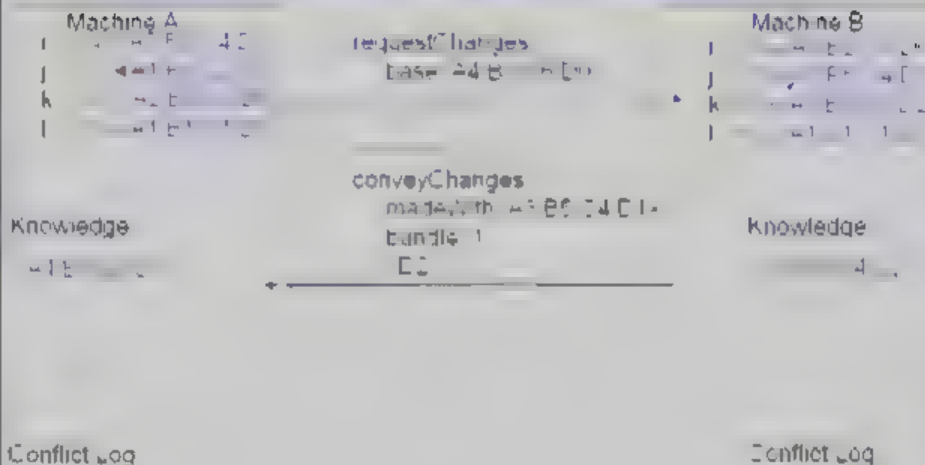
Conflict Log

Conflict Log

Protocol with Conflict Handling



Protocol with Conflict Handling



Protocol with Conflict Handling

Machine A

I: A4 - A1 B1 C1 D0
 J: C2 - A3 B5 C4 D1
 K: C5 - A2 B3 C4 D0
 L: A2 - A1 B1 C1 D0

requestChanges

base: A4 B1 C1 D0

Machine B

I: A4 - A1 B1 C1 D0
 J: C2 - A3 B5 C4 D1
 K: B5 - A3 B4 C4 D2
 L: A2 - A1 B1 C1 D0

conveyChanges

makeup, the A3 B4 C4 D1

bundle 1

J D2

Knowledge

A3 B5 C4 D2

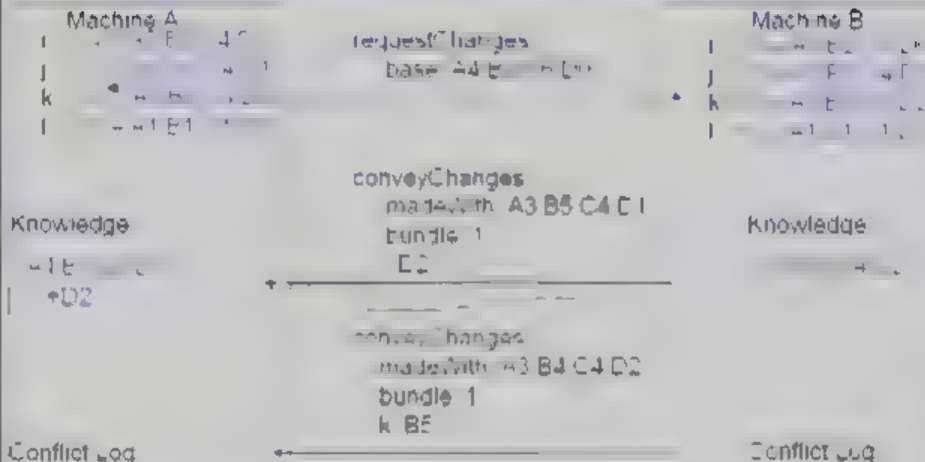
Knowledge

A4 B3 C6 D0
 + D2

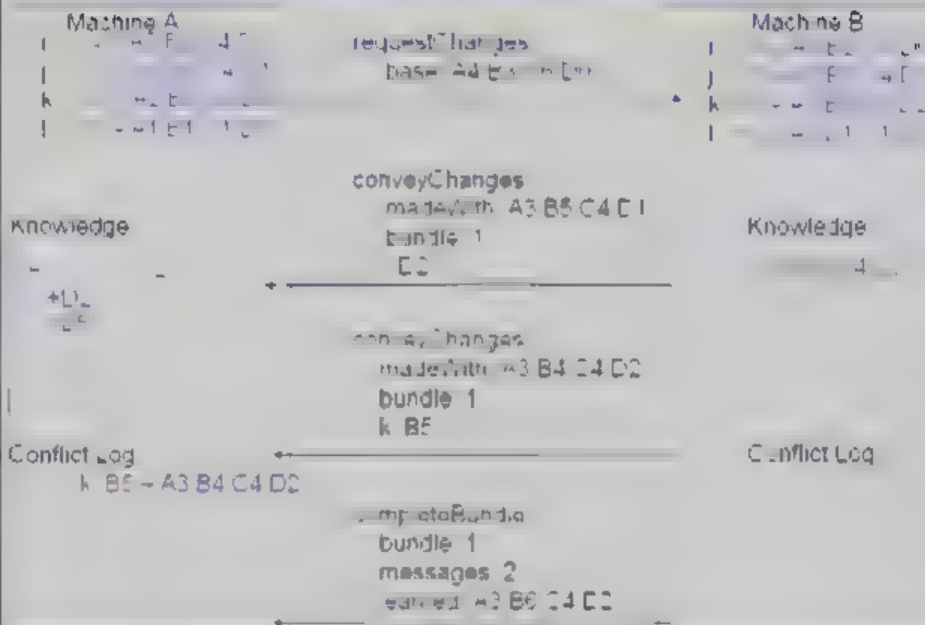
Conflict Log

Conflict Log

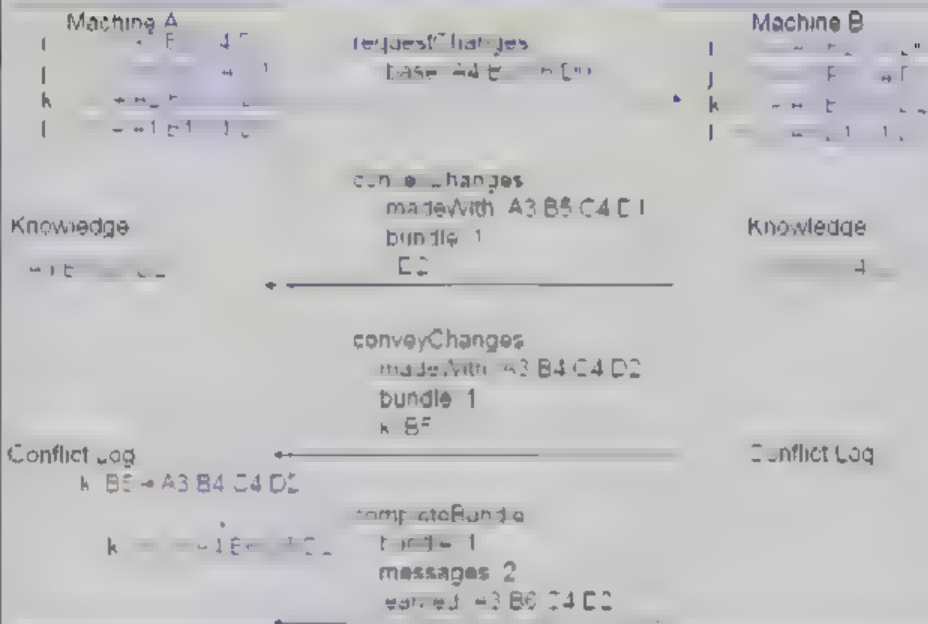
Protocol with Conflict Handling



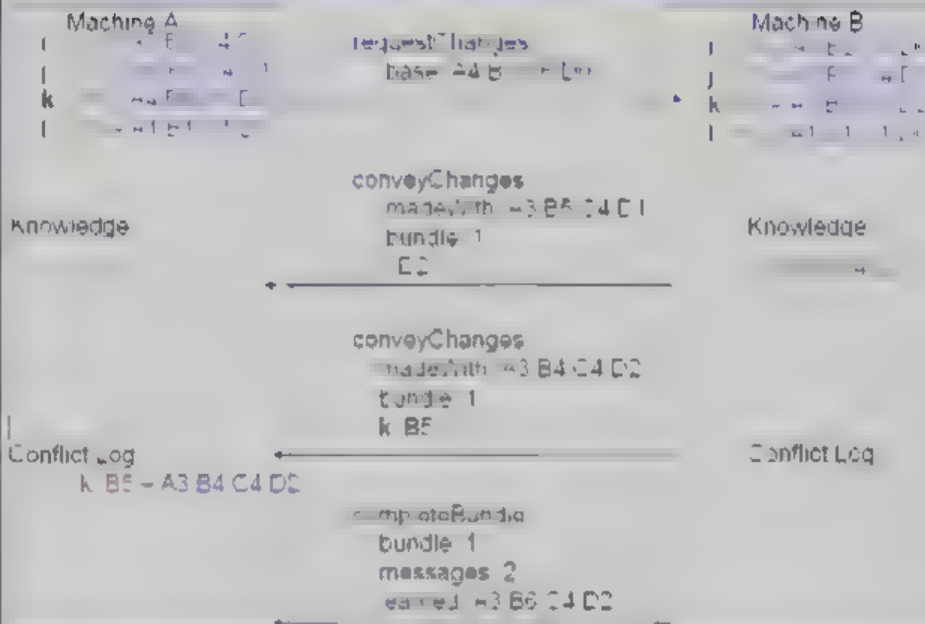
Protocol with Conflict Handling



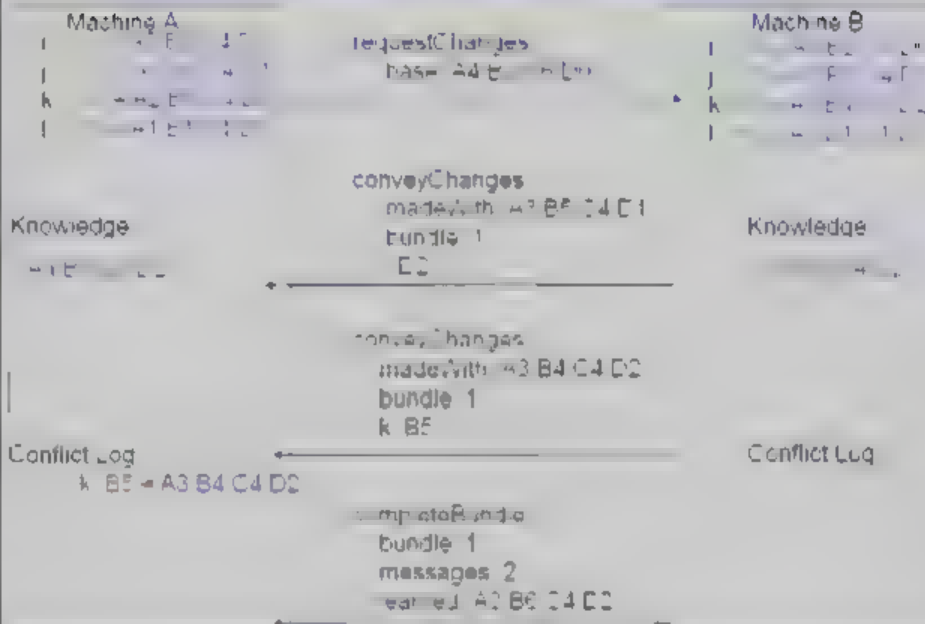
Protocol with Conflict Handling



Protocol with Conflict Handling



Protocol with Conflict Handling



Protocol with Conflicts: Not Shown

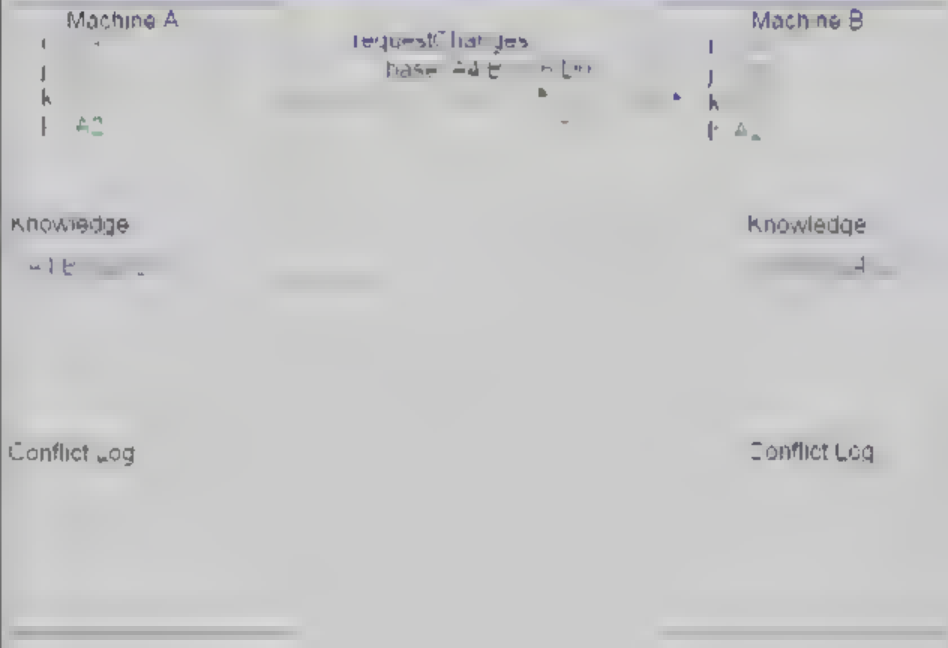
- Automatic Conflict Resolution

- If a node receives a message from a node that is not its parent, it must check if the message is a duplicate of a message it has already received.
- If the message is a duplicate, it must be discarded.
- If the message is not a duplicate, it must be added to the node's message set.
- If the message is a duplicate of a message that has already been received from a different node, it must be discarded.

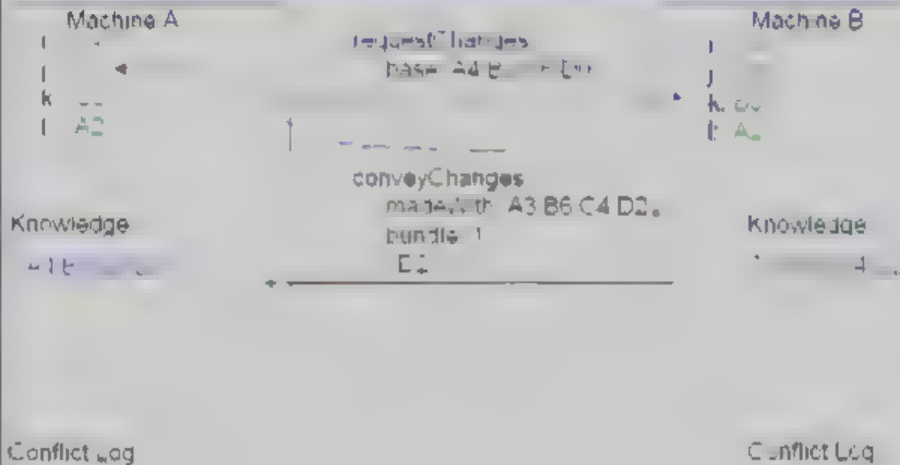
- Storage Optimization

- If a node receives a message from a node that is not its parent, it must check if the message is a duplicate of a message it has already received.
- If the message is a duplicate, it must be discarded.
- If the message is not a duplicate, it must be added to the node's message set.

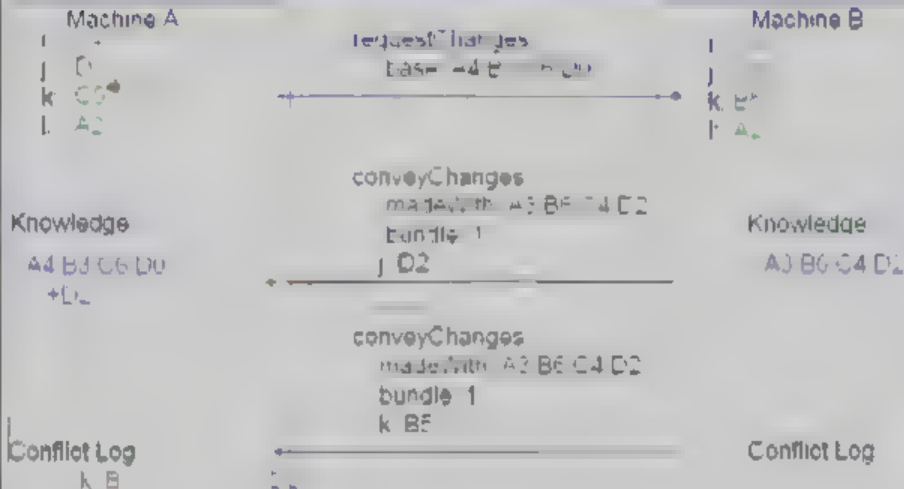
Protocol with Implicit "Made With" Knowledge



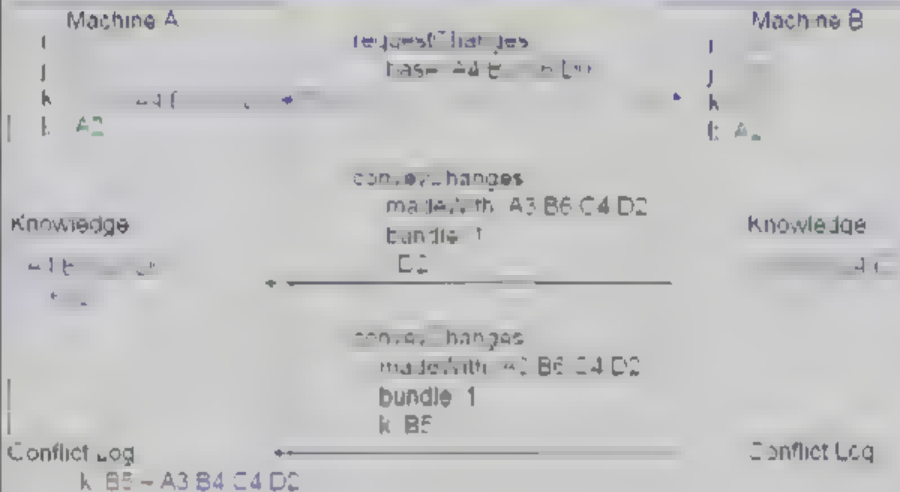
Protocol with Implicit "Made With" Knowledge



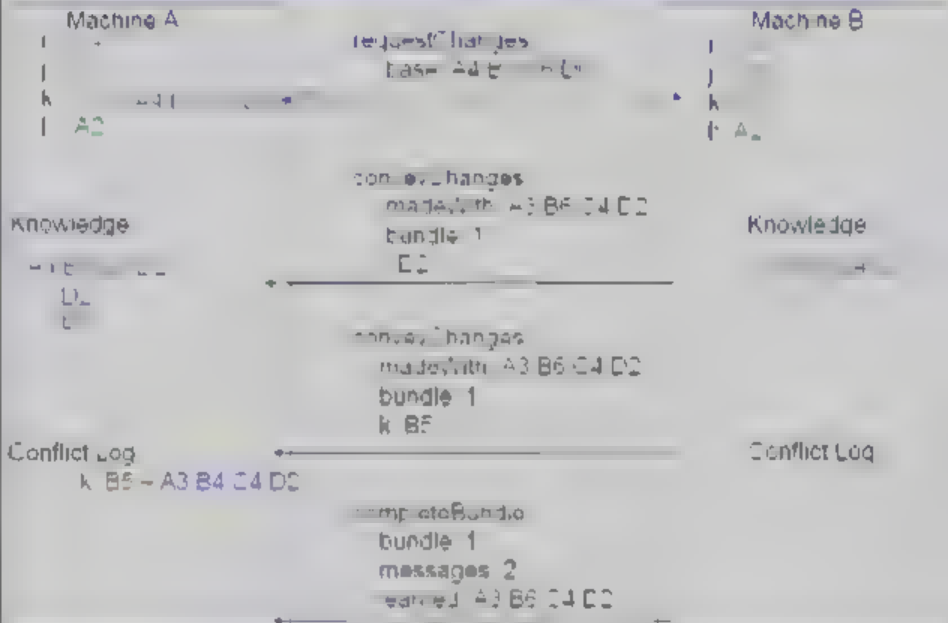
Protocol with Implicit "Made With" Knowledge



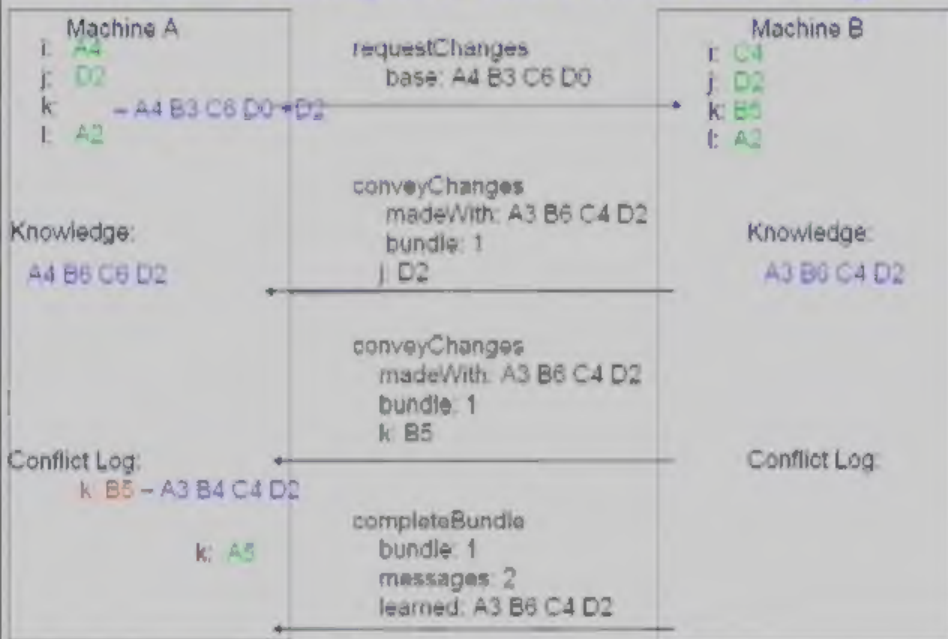
Protocol with Implicit "Made With" Knowledge



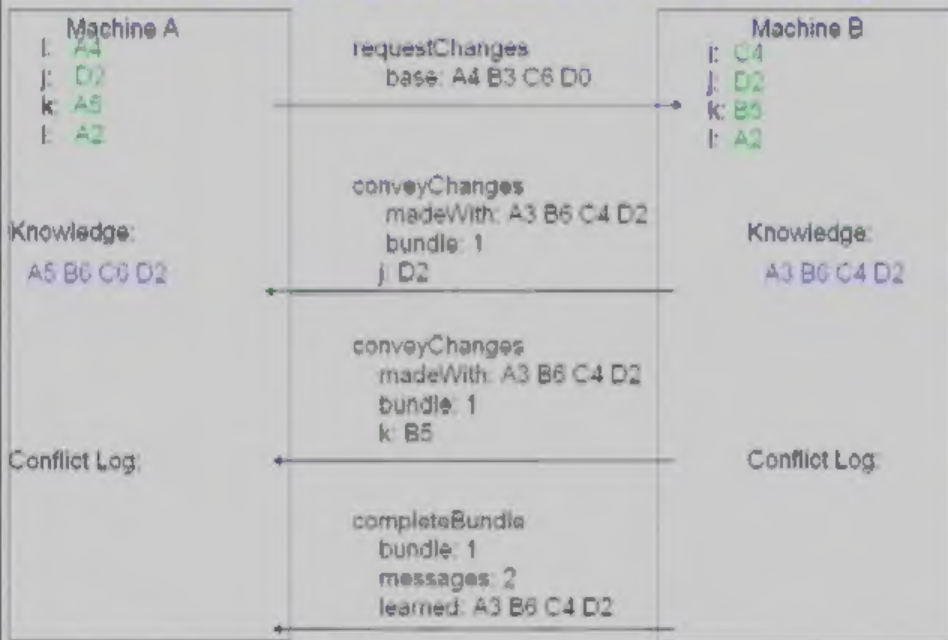
Protocol with Implicit "Made With" Knowledge



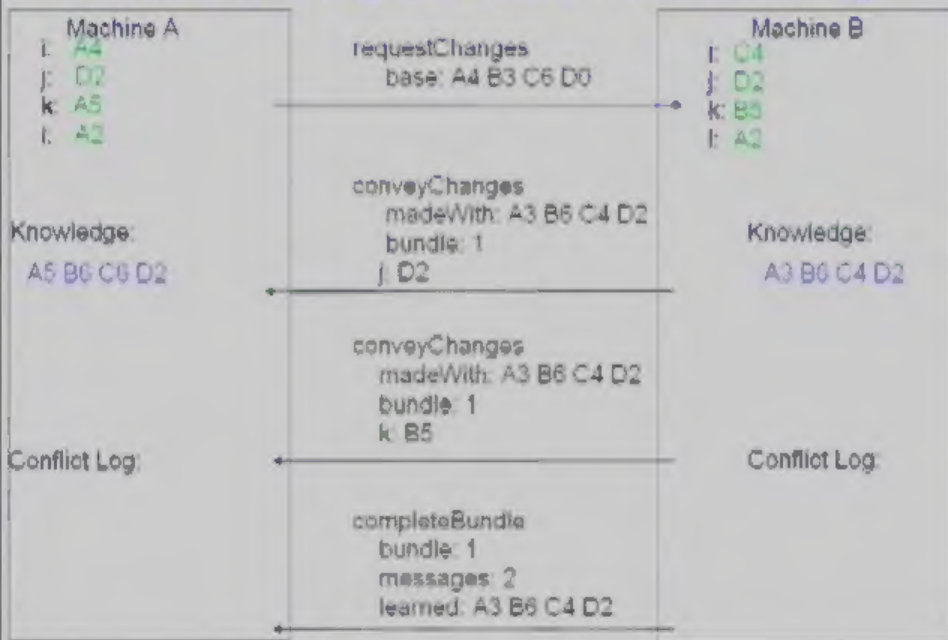
Protocol with Implicit "Made With" Knowledge



Protocol with Implicit "Made With" Knowledge



Protocol with Implicit "Made With" Knowledge



Summary

Key features of WinFS Synchronization:

- Favors availability over consistency
- Achieves robustness and scalability through pair-wise, knowledge-driven protocol
- Makes efficient use of bandwidth and storage
- Allows arbitrary communication topology
- Guarantees eventual convergence if well-connected topology
- Detects conflicts; permits manual or automatic resolution

